ALTERNATING PROXIMAL GRADIENT METHOD FOR NONNEGATIVE MATRIX FACTORIZATION

YANGYANG XU *

Abstract. Nonnegative matrix factorization has been widely applied in face recognition, text mining, as well as spectral analysis. This paper proposes an alternating proximal gradient method for solving this problem. With a uniformly positive lower bound assumption on the iterates, any limit point can be proved to satisfy the first-order optimality conditions. A Nesterov-type extrapolation technique is then applied to accelerate the algorithm. Though this technique is at first used for convex program, it turns out to work very well for the non-convex nonnegative matrix factorization problem. Extensive numerical experiments illustrate the efficiency of the alternating proximal gradient method and the accleration technique. Especially for real data tests, the accelerated method reveals high superiority to state-of-the-art algorithms in speed with comparable solution qualities.

1. Introduction. Given a nonnegative matrix M, nonnegative matrix factorization (NMF) aims to find a pair of nonnegative factors X, Y such that M is approximated by the product of X and Y. Mathematically, this problem can be modeled as

$$\min_{X,Y} F(X,Y) \equiv \frac{1}{2}\|XY - M\|_F^2, \text{ subject to } X \in \mathbb{R}_+^{m\times r}, Y \in \mathbb{R}_+^{r\times n}, \tag{1.1}$$

where M is the given $m \times n$ nonnegative matrix and r (usually $r \ll \min\{m,n\}$) is pre-determined. It appears that NMF was first proposed and used by Paatero and his coworkers in areas of environmental science [17]. Unfortunately, they used "*positive* matrix factorization" in their paper though they actually considered "*nonnegative* matrix factorization". The later popularity of NMF attributes a great deal to the publication of [10] in *nature*. Since then, NMF has been widely applied in data mining such as text mining [19], image mining [12], dimension reduction and clustering [3, 22], as well as spectral data analysis [18]. More information on NMF can be found in the survey paper [2], books [4, 5] and the references thereof. Recently, some variants of NMF were proposed in order to better fit applications, such as semi-NMF [6] and convex-NMF [20]. Discussions of these models have been out of the scope of this paper. We refer the interested readers to the papers where they are proposed and used.

1.1. Existing algorithms for NMF. Since problem (1.1) is non-convex and it is proved NP-hard, no algorithm can be found to exactly solve the problem within polynomial time unless P=NP. Two early widely used algorithms for NMF are projected alternating least squares method [17] and multiplicative updating method [11]. The former algorithm alternatively updates X and Y by minimizing $\|XY - M\|_F^2$ with respect to X and Y without the nonnegativity constraint and then projecting the solution to the nonnegative matrix cone. The closed-form updates are given as

$$\begin{aligned} X^k &= \mathcal{P}_+(M(Y^{k-1})^\top (Y^{k-1}(Y^{k-1})^\top)^\dagger), \\ Y^k &= \mathcal{P}_+(((X^k)^\top X^k)^\dagger (X^k)^\top M), \end{aligned}$$

where $(\mathcal{P}_+(Z))_{ij} = \max(0, Z_{ij})$ and $\dagger$ denotes pseudo-inverse. The latter algorithm has much cheaper multiplicative updates

$$\begin{aligned} (X^k)_{ij} &= (X^{k-1})_{ij}(M(Y^{k-1})^\top)_{ij}/(X^{k-1}Y^{k-1}(Y^{k-1})^\top + \varepsilon)_{ij}, \ \forall i,j, \\ (Y^k)_{ij} &= (Y^{k-1})_{ij}((X^k)^\top M)_{ij}/((X^k)^\top (X^k)Y^{k-1} + \varepsilon)_{ij}, \ \forall i,j \end{aligned}$$

where $\varepsilon > 0$ is used to avoid *zero* division. This method is simple to implement and often yields good results. However, it still lacks of convergence result, and it is quite sensitive to initial values. Once one component of X or Y becomes *zero* at some iteration, it will stay *zero* during the whole process of the method.

Despite of joint non-convexity, the objective function is convex with respect to each variable by fixing the other. Based on this observation, a series of alternating nonnegative least square methods (ANLS) were proposed such as projected gradient method [13], active set method [8] and block principal pivoting method [9]. These methods alternatively minimize the objective function F with respect to X and Y, one at a time

*yangyang.xu@rice.edu. Dept. of Applied and Computational Mathematics, Rice University, Houston, TX.

while fixing the other at its most recent value. Both X-subproblem and Y-subproblem can be written as the following nonnegative least square problem

$$\min_{W\geq 0} h(W) \equiv \frac{1}{2}\|AW - B\|_F^2. \tag{1.2}$$

Therefore, the speed of ANLS-type methods largely depend on how fast problem (1.2) can be solved. Recently, an ADM-based algorithm [23] was proposed for solving problem (1.1) and was observed superior to many other algorithms for both synthetic data and real image data. Due to non-convexity, the classical ADM is not guaranteed to converge. Hence, only a preliminary convergence result was provided under a strong assumption that the difference of two consecutive iterates converges to *zero*. Compared to ANLS-type methods, the ADM-based algorithm maintains faster convergence because it does not solve (1.2) exactly but instead makes one ADM update during each iteration. Unlike ANLS-type methods, the ADM-based algorithm is not guaranteed to decrease the objective function value after every iteration. In this paper, we will propose an alternating proximal gradient method for solving (1.1). This method maintains fast convergence like ADM-based algorithm but also has a better convergence result.

1.2. Organization. The rest of this paper is organized as follows. An alternating proximal gradient method for solving (1.1) is derived and analyzed in Section 2. In order to speed up the algorithm, a Nesterov-type acceleration technique is applied. Extensive numerical experiments are done in Section 3. It illustrates that the applied acceleration technique indeed speeds up the proposed algorithm a lot. In addition, fast convergence is observed by comparing to several other existing methods. Especially for real data tests, our method is significantly better than the other compared algorithms. Section 4 concludes this paper by extending the same method to a sparse regularized NMF model.

2. Algorithm and convergence result. Though ANLS-type method spends much time solving subproblem (1.2), it converges within quite a few outer iterations. The fast outer convergence can probably be explained that a sufficient decrease of the objective function is obtained after each iteration. In this section, we will propose an algorithm which will obtain a sufficient decrease of the objective function after every iteration but never spends too much on the subproblem (1.2). Therefore, it will maintain an overall faster convergence than ANLS-type methods.

2.1. Proximal gradient method. Consider the convex optimization problem

$$\min_{x\in\mathcal{X}} f(x), \tag{2.1}$$

where $\mathcal{X}$ is a convex set and f is a *smooth* convex function with Lipschitz continuous gradient, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L_f\|x - y\|_2, \quad \text{for all } x, y \in \mathcal{X},$$

where L_f is the Lipschitz constant. The projected gradient method for (2.1) is

$$x^{k+1} = \mathcal{P}_{\mathcal{X}}(x^k - \alpha_k \nabla f(x^k)).$$

Choosing appropiate stepsize α_k, the new iteration point x^{k+1} can be regarded as a minimizer of the linearized proximal regularization of function f at point x^k, i.e.,

$$x^{k+1} = \arg\min_{x\in\mathcal{X}} f(x^k) + \langle \nabla f(x^k), x - x^k\rangle + \frac{L_f}{2}\|x - x^k\|_2^2.$$

In general, we can update the iterate by letting

$$x^{k+1} = \arg\min_{x\in\mathcal{X}} f(y^k) + \langle \nabla f(y^k), x - y^k\rangle + \frac{L_f}{2}\|x - y^k\|_2^2, \tag{2.2}$$

where y^k is a given point at which f is linearized. The closed form solution of (2.2) is given by

$$x^{k+1} = \mathcal{P}_{\mathcal{X}}(y^k - \nabla f(y^k)/L_f).$$

With appropriate choice of y^k, this method is guaranteed to reach a solution of (2.1). Simply letting $y^k = x^k$, an ε-optimal solution can be obtained within $O(1/\varepsilon)$ iterations. With a clever choice of y^k, an ε-optimal solution can be reached within $O(1/\sqrt{\varepsilon})$ iterations. For details, please refer to [1, 16].

We can use this method to solve subproblem (1.2) by repeatedly updating W by

$$W^{k+1} = \max(0, W^k - \nabla h(W^k)/L_h), \tag{2.3}$$

until convergence, where L_h is a Lipschitz constant of gradient $\nabla h(W)$. However, as mentioned above, it may take much time to exactly solve the subproblems at each iteration. It turns out that it is sufficient to implement one step of update (2.3) for both X-subproblem and Y-subproblem at each iteration. Namely, we can update X, Y by

$$X^{k+1} = \max(0, X^k - \nabla_X F(X^k, Y^k)/L_{X^k}), \tag{2.4a}$$
$$Y^{k+1} = \max(0, Y^k - \nabla_Y F(X^{k+1}, Y^k)/L_{Y^k}), \tag{2.4b}$$

where L_{X^k} and L_{Y^k} are Lipschitz constants of gradient $\nabla_X F(X, Y^k)$ and $\nabla_Y F(X^{k+1}, Y)$ by regarding F as a function with respect to X and Y, respectively. In our algorithm, we will take L_{X^k} and L_{Y^k} as the spectral norm of $Y^k(Y^k)^\top$ and $(X^{k+1})^\top X^{k+1}$, respectively. Since $r \ll \min\{m, n\}$, this can be done very efficiently at each iteration.

The pseudocode of the above algorithm can be written as follows.

Algorithm 1 Alternating proximal gradient method for NMF (APG)

Input: $m \times n$ nonnegative matrix M and pre-determined dimension r
Initialization: randomize X^0 and Y^0 as nonnegative matrices of appropriate size
for $k = 0, 1, 2, \ldots$ **do**
  Update X, Y according to formula (2.4).
  if some stopping criterion is met **then**
    Stop and output (X^{k+1}, Y^{k+1})
  end if
end for

2.2. Convergence analysis. The update method (2.2) is analyzed in [1] with $\mathcal{X} = \mathbb{R}^n$. However, it is mentioned that their main result Lemma 2.3 can be adapted for the case with convex constriant $x \in \mathcal{X}$. Therefore, we can get the following result through essentially the same proof of Lemma 2.3 in [1] which we omit here.

LEMMA 2.1. *Let y^k be a given point in the domain of f and L_f be a Lipschitz constant of ∇f. Suppose x^{k+1} is obtained by (2.2). Then for any $x \in \mathcal{X}$,*

$$f(x) - f(x^{k+1}) \geq \frac{L_f}{2}\|x^{k+1} - y^k\|_2^2 + L_f\langle y^k - x, x^{k+1} - y^k\rangle.$$

Specifically, for updates (2.4), it holds for any nonnegative matrices X, Y that

$$F(X, Y^k) - F(X^{k+1}, Y^k) \geq \frac{L_{X^k}}{2}\|X^k - X^{k+1}\|_F^2 + L_{X^k}\langle X^k - X, X^{k+1} - X^k\rangle,$$

and

$$F(X^{k+1}, Y^k) - F(X^{k+1}, Y^{k+1}) \geq \frac{L_{Y^k}}{2}\|Y^k - Y^{k+1}\|_F^2 + L_{Y^k}\langle Y^k - Y, Y^{k+1} - Y^k\rangle.$$

Based on this lemma, we next provide a convergence result of algorithm APG under some mild assumptions. A point (X, Y) satisfies the KKT-condition of (1.1) if

$$X \odot (XY - M)Y^\top = 0, \quad X \geq 0, \quad (XY - M)Y^\top \geq 0 \tag{2.5a}$$
$$Y \odot X^\top(XY - M) = 0, \quad Y \geq 0, \quad X^\top(XY - M) \geq 0 \tag{2.5b}$$

where $\odot$ denotes component-wise product.

THEOREM 2.2. *Suppose the sequence $\{(X^k, Y^k)\}$ generated by algorithm APG is uniformly away from* zero*, i.e., there exists $L > 0$ such that $L_{X^k} \geq L$ and $L_{Y^k} \geq L$. Then any limit point of $\{(X^k, Y^k)\}$ satisfies the KKT-conditions* (2.5).

Proof. Replacing X with X^k, Y with Y^k in Lemma 2.1, we obtain that

$$F(X^k, Y^k) - F(X^{k+1}, Y^{k+1}) \geq \frac{L_{X^k}}{2}\|X^{k+1} - X^k\|_F^2 + \frac{L_{Y^k}}{2}\|Y^{k+1} - Y^k\|_F^2 \tag{2.6}$$

which means the objective function value $F(X^k, Y^k)$ is nonincreasing. Since $F(X^k, Y^k)$ is bounded below by *zero*, then it must converge to some real number $\bar{F} \geq 0$. Summing up both sides of inequality (2.6), we have

$$\begin{aligned} F(X^0, Y^0) - \bar{F} &\geq \sum_k \left(\frac{L_{X^k}}{2}\|X^{k+1} - X^k\|_F^2 + \frac{L_{Y^k}}{2}\|Y^{k+1} - Y^k\|_F^2 \right) \\ &\geq \frac{L}{2}(\|X^{k+1} - X^k\|_F^2 + \|Y^{k+1} - Y^k\|_F^2), \end{aligned}$$

where the last inequality follows from $L_{X^k} \geq L$ and $L_{Y^k} \geq L$. Therefore, the differences $X^{k+1} - X^k$ and $Y^{k+1} - Y^k$ both asymptotically vanish.

Suppose $(\bar{X}, \bar{Y})$ is a limit point of $\{(X^k, Y^k)\}$. Then there exists a subsequence $\{(X^{k_j}, Y^{k_j})\}$ converging to $(\bar{X}, \bar{Y})$. Based on the above analysis, the sequence $\{(X^{k_j+1}, Y^{k_j+1})\}$ also converges to $(\bar{X}, \bar{Y})$. According to the update rules (2.4), it holds that

$$\begin{aligned} X^{k_j+1} &= \max(0, X^{k_j} - (X^{k_j}Y^{k_j} - M)(Y^{k_j})^\top / L_{X^{k_j}}), \\ Y^{k_j+1} &= \max(0, Y^{k_j} - (X^{k_j+1})^\top (X^{k_j+1}Y^{k_j} - M)/L_{Y^{k_j}}). \end{aligned}$$

Letting $j \to \infty$, we have

$$\begin{aligned} \bar{X} &= \max(0, \bar{X} - (\bar{X}\bar{Y} - M)\bar{Y}^\top / L_{\bar{X}}), \\ \bar{Y} &= \max(0, \bar{Y} - \bar{X}^\top (\bar{X}\bar{Y} - M)/L_{\bar{Y}}), \end{aligned}$$

where $L_{\bar{X}} = \lambda_{\max}(\bar{Y}\bar{Y}^\top)$ and $L_{\bar{Y}} = \lambda_{\max}(\bar{X}^\top\bar{X})$. This implies $\bar{X}$ is a minimizer of the linearized proximal regularization of $F(X, \bar{Y})$ regarded as a function with respect to X at point $\bar{X}$, i.e.,

$$\bar{X} = \arg\min_{X \geq 0} F(\bar{X}, \bar{Y}) + \langle \nabla_X F(\bar{X}, \bar{Y}), X - \bar{X} \rangle + \frac{L_{\bar{X}}}{2}\|X - \bar{X}\|_F^2.$$

Assume X^* is a solution of problem

$$\min_{X \geq 0} F(X, \bar{Y}). \tag{2.7}$$

Using Lemma 2.1 again, we have $F(X^*, \bar{Y}) - F(\bar{X}, \bar{Y}) \geq 0$, which implies $\bar{X}$ is also a solution of (2.7). Therefore, $\bar{X}$ must satisfy the KKT-conditions of (2.7) which is exactly (2.5a). In the same way, we can prove $(\bar{X}, \bar{Y})$ satisfies (2.5b). This completes the proof. □

2.3. Accelerated proximal gradient method. Though the proximal gradient method maintains a nice convergence result, it converges relatively slow as it approaches the solution. To speed up the algorithm, an extrapolation technique can be applied to accelerate its convergence. Back to the update (2.2), we can linearize the function f at $y^k = x^k + \omega_k(x^k - x^{k-1})$ instead of $y^k = x^k$, where $\omega_k \geq 0$ is extrapolation weight. The intuitive idea behind this kind of acceleration technique can be explained as follows. If the current iterate x^k approaches the optimal point x^* more than the previous iterate x^{k-1}, then the extrapolation point may be closer to x^* than x^k with appropriate extrapolation weight. See Figure 2.1 for illustration. For convex program, specific extrapolation weight can be chosen such that overall acceleration can be reached. For example, $w_k = (t_{k-1} - 1)/t_k$ is used and analyzed in [1, 16], where $t_k = (1 + \sqrt{1 + 4t_{k-1}^2})/2$ with initial value $t_0 = 1$. This technique called Nesterov-type acceleration technique makes the proximal gradient method

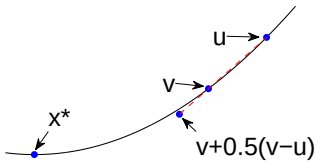


FIG. 2.1. *u is the previous iterate and v is the current iterate. The extrapolation point $v + 0.5(v - u)$ becomes closer to the optimal point x^* than v.*

reach a theoretically best convergence rate of first-order method [15]. It is also observed that an appropriate extrapolation works well for non-convex problem. The authors of [21, 14] dynamically chose a sequence of extrapolation weights which makes their SOR method significantly outperform the corresponding Gauss Seidel method.

In this section, we will use Nesterov-type acceleration technique to speed up algorithm APG, since essentially the algorithm APG alternatively applies proximal gradient method to the subproblems. It turns out that the acceleration technique can indeed speed up algorithm APG. Actually, the heuristic extrapolation technique used in [21, 14] is also helpful for our algorithm, but not so stable as the Nesterov-type technique for problem (1.1). Currently, we are not able to provide a theoretical analysis about why the Nesterov-type acceleration technique works so well for non-convex problem. We reserve it as our future work. Incorporating the acceleration technique into algorithm APG, we arrive at the following accelerated APG method.

Algorithm 2 Accelerated alternating proximal gradient method for NMF (AAPG)

Input: $m \times n$ nonnegative matrix M and pre-determined dimension r
Initialization: randomize X^{-1} and Y^{-1} as nonnegative matrices of appropriate size. Set $X^0 = X^{-1}$, $Y^0 = Y^{-1}$ and $t_{-1} = 1$.
for $k = 0, 1, 2, \ldots$ **do**

$$t_k = (1 + \sqrt{1 + 4t_{k-1}^2})/2 \tag{2.8a}$$
$$\omega_k = (t_{k-1} - 1)/t_k \tag{2.8b}$$
$$Z_X^k = X^k + \omega_k(X^k - X^{k-1}) \tag{2.8c}$$
$$Z_Y^k = Y^k + \omega_k(Y^k - Y^{k-1}) \tag{2.8d}$$
$$X^{k+1} = \max(0, Z_X^k - \nabla_X F(Z^k, Y^k)/L_{X^k}), \tag{2.8e}$$
$$Y^{k+1} = \max(0, Z_Y^k - \nabla_Y F(X^{k+1}, Z_Y^k)/L_{Y^k}), \tag{2.8f}$$

  if some stopping criterion is met **then**
    Stop and output (X^{k+1}, Y^{k+1})
  end if
end for

To make the objective function value decreasing after each iteration, we can let $(X^{k+1}, Y^{k+1}) = (X^k, Y^k)$ and $t_k = t_{k-1}$, whenever $F(X^{k+1}, Y^{k+1}) > F(X^k, Y^k)$. Throughout all of our tests, we will apply the above modification to algorithm AAPG. However, it performs very robust even without such modification. It is observed that the above situation only happens when r becomes relatively large. Figure 2.2 plots the comparison results between proximal gradient method and its accelerated version. The tested matrix is in the form of $M = LR + \sigma N$, where L and R are generated by Matlab command `randn(m,r)` and `randn(r,n)` and then projected to nonnegative matrix cone, respectively, and N is a nonnegative Gaussian noise. In this test, m and n are both set to 300, r is set to 20, and noise-level $\sigma = 0.05$ is chosen for the noise case. The relative error is calculated by $\|XY - M\|_F / \|M\|_F$. We can see that the convergence speed of APG is

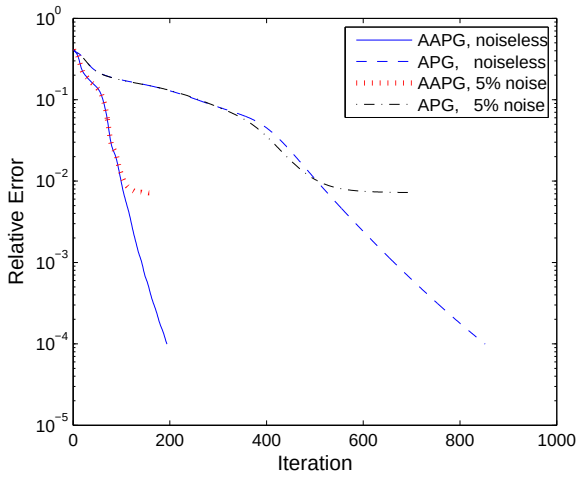


FIG. 2.2. *Comparison results for algorithm APG and AAPG with random matrix M of size* 300×300.

increased at least 3 times with acceleration technique for both noiseless and noise cases.

3. Numerical experiments. In this section, we will compare the algorithms proposed in last section and some state-of-the-art algorithms with both synthetic data and real data. So far, there have been a great quantity of algorithms proposed for solving NMF. We do not want to exhaust all these methods but only consider several most popular ones or most recent ones that have been shown superior to many other algorithms. The first two we will compare with are alternating least square method (Als) [17, 2] and multiplicative updating method (mult) [11]. Both of the methods have been written as Matlab functions and can be called by `nnmf` with solver specifier `als` and `mult`, respectively. One recent ANLS-type method blockpivot is included for comparison. It is shown in [9] that blockpivot outperforms all the other compared ANLS-type methods in both speed and solution quality. Another algorithm we want to compare with is the recent ADM-based method (ADM) introduced in [23]. Although both blockpivot and ADM have been shown superior to Als and mult in many cases, we still include Als and mult for comparison due to their popularity and easy availability.

3.1. Implemention details. From inequality (2.6), it follows that the objective function value is nonincreasing. The small change between two consecutive objective function values implies the closeness between two iterates. Hence, we will terminate algorithm APG and AAPG whenever

$$\frac{F_k - F_{k+1}}{F_k + 1} \leq tol,$$

where F_k denotes $F(X^k, Y^k)$ and tol is a given tolerance. The algorithms are also terminated if a relatively small objective function value has been reached, i.e.,

$$\frac{F_k}{\|M\|_F} \leq tol.$$

In addition, we require the first conditions to be satisfied in at least *three* consecutive iterations. Here, we simply use a same tolerance for both stopping criterions. Of course, we can apply two diffenrent tolerances for diffenrent conditions.

Throughout all of our tests, we will set $tol = 10^{-4}$ for all algorithms except ADM since 10^{-4} is somehow loose for ADM. Hence, we set $tol = 10^{-5}$ for ADM. The maximum number of iterations is set to 1000 for all algorithms. Same initial values are used for all the methods except mult since mult is very sensitive to initial value. Instead, we let mult replicatively do 10 times with a maximum number of iterations set to 10 at the beginning and then choose the best output as its initial value. For all the compared methods, all parameters are set to their default values and default termination criterions are employed, unless specified otherwise. All tests are performed on a Lenovo T410 laptop with an i7-620m CPU and 3-GB RAM and running 32-bit Windows 7 and Matlab 2010b.

TABLE 3.1
Comparison results with nonnegative Gaussian random matrices of size $m \times n$ with n fixed at 1000.

	APG		AAPG		ADM		blockpivot		Als		mult	
(m, r)	Relerr	Time	Relerr	Time	Relerr	Time	Relerr	Time	Relerr	Time	Relerr	Time
(200, 10)	9.85e-5	0.60	9.42e-5	0.21	8.41e-5	0.28	8.83e-5	0.87	7.93e-5	0.47	1.10e-2	2.37
(200, 20)	1.11e-4	2.89	9.76e-5	0.75	1.41e-4	0.68	1.06e-4	3.40	7.46e-5	0.83	1.20e-2	5.88
(200, 30)	1.46e-2	5.02	9.83e-5	1.63	1.94e-4	1.19	1.15e-4	6.45	7.57e-5	1.40	2.24e-2	11.4
(1000, 10)	1.00e-4	2.53	9.23e-5	0.93	6.50e-5	1.10	3.87e-5	1.58	5.42e-5	2.38	9.81e-3	11.3
(1000, 20)	9.98e-5	10.5	9.39e-5	2.58	1.09e-4	2.20	7.65e-5	5.74	3.71e-5	5.51	1.04e-2	26.5
(1000, 30)	2.22e-3	20.4	9.86e-5	5.83	1.65e-4	3.97	9.11e-5	10.8	3.76e-5	8.92	1.26e-2	50.5

TABLE 3.2
Comparison results on 2000 selected images from CBCL face database with 361×2000 *tested matrix and dimension* r *varying among* $\{30, 60, 90\}$

	AAPG		ADM		blockpivot		Als		mult	
r	Relerr	Time	Relerr	Time	Relerr	Time	Relerr	Time	Relerr	Time
30	1.91e-1	3.68	1.92e-1	7.33	1.90e-1	21.5	3.53e-1	3.15	2.13e-1	6.51
60	1.42e-1	12.5	1.43e-1	19.5	1.40e-1	63.2	4.59e-1	1.80	1.74e-1	12.1
90	1.13e-1	26.7	1.15e-1	34.2	1.12e-1	111	6.00e-1	2.15	1.52e-1	18.4

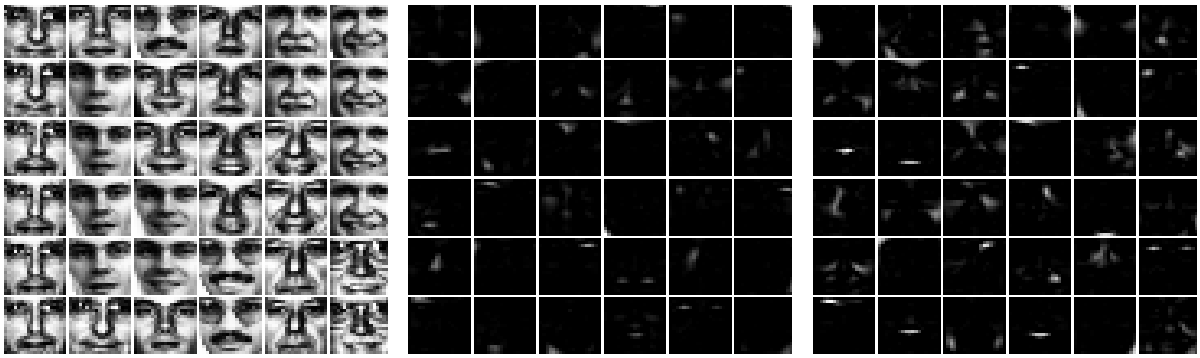
3.2. Random matrices. Nonnegative Gaussian random matrices are employed in this subsection for comparison. Each matrix can be written as $M = LR$ where $L \in \mathbb{R}_+^{m\times r}$ and $R \in \mathbb{R}_+^{r\times n}$, both generated by Matlab command `randn` and then projected to nonnegative matrix cone. Note that the dimension r used here is exactly same as that used in the model (1.1). Namely, each tested matrix can be exactly decomposed as the product of two low dimensional nonnegative matrices and we use exact rank estimation in this subsection. Algorithms are compared with fixed $n = 1000$ and m chosen from $\{200, 1000\}$, r from $\{10, 20, 30\}$. Table 3.1 lists the detailed results. Each result is average of 10 independent trails. Relerr denotes relative error calculated by $\|XY - M\|_F/\|M\|_F$, and time is measured by CPU time (sec). From the table, we can see that all methods get quite good solutions except APG for $r = 30$ and mult for all r's. It is observed that APG did not stop within 1000 iterations for $r = 30$. That is why APG outputs worse solutions at $r = 30$. AAPG and ADM are comparable in both CPU time and solution quality. Blockpivot and Als reach more accurate solutions most times but take more time than AAPG and ADM. Since AAPG is always better than APG, we will drop APG from our comparison list in the next subsections.

3.3. Image data. In this subsection, we will test algorithm AAPG, ADM, blockpivot, Als and mult on two image databases used in [7, 13]:

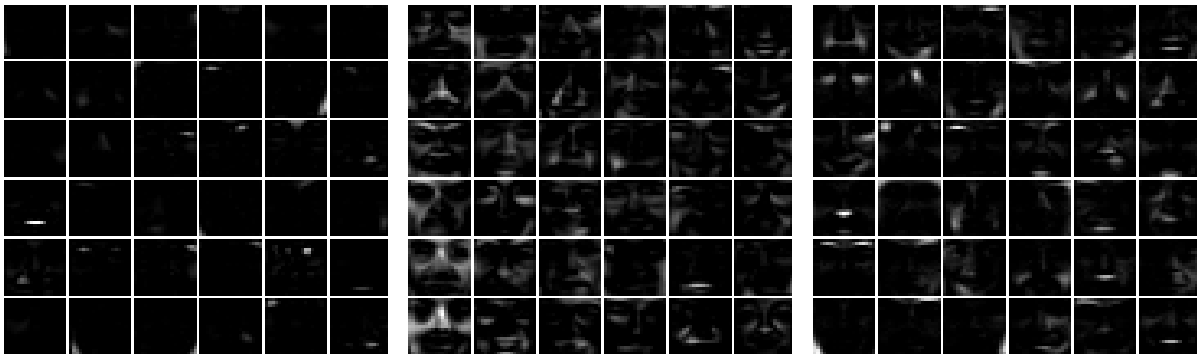
- CBCL database (http://www.ai.mit.edu/projects/cbcl)
- ORL database (http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html)

There are 6977 face images in the training set of CBCL database, with each one of 19×19 pixels. For our test, 2000 images are selected from the training set. We reshape each image into a column vector and obtain a matrix M of size 361×2000. Figure 3.1 (a) shows the first 36 faces corresponding to the first 36 columns of matrix M. Each face is photoed more than once with difference illuminations or expressions. In this test, we vary r in $\{30, 60, 90\}$. The comparison results are shown in Table 3.2. All results are average of 10 independent trials. From the table, we can see that AAPG outperforms ADM in both time and solution quality. Blockpivot gets slightly better solutions than AAPG (less than 1% accurate), but is much slower (about 400% more time). Algorithm Als and mult actually both fail in this test. Especially for Als, it stagnates at a low-quality solution very early. Figure 3.1 (b)-(f) plots 36 base images corresponding to the first 36 columns of basis matrix X for each method at $r = 90$. Each basis matrix X is normalized such that the maximum element equals to *one* before plotting. As shown in [10], NMF can be used to learn parts of an object. The successful algorithms AAPG, ADM and blockpivot all get basis matrices containing local features of face images. All of the three basis matrices are relatively sparse. The basis matrix X obtained by AAPG is the sparsest one with about 24.9% non-zeros after zeroing out all elements below 10^{-6}. There are about 31.5% non-zeros for the densest one obtained by ADM. Due to the poor performance of Als and mult, we will only consider AAPG, ADM and blockpivot in the following tests to save testing time.

FIG. 3.1. *CBCL database and comparison base images: (a) 36 images selected from 2000 tested images; (b)-(f) 36 base images corresponding to the first 36 columns of basis matrix X obtained from algorithm AAPG, ADM, blockpivot, Als and mult at $r = 90$.*



(a) CBCL data (b) AAPG (c) ADM



(d) blockpivot (e) Als (f) mult

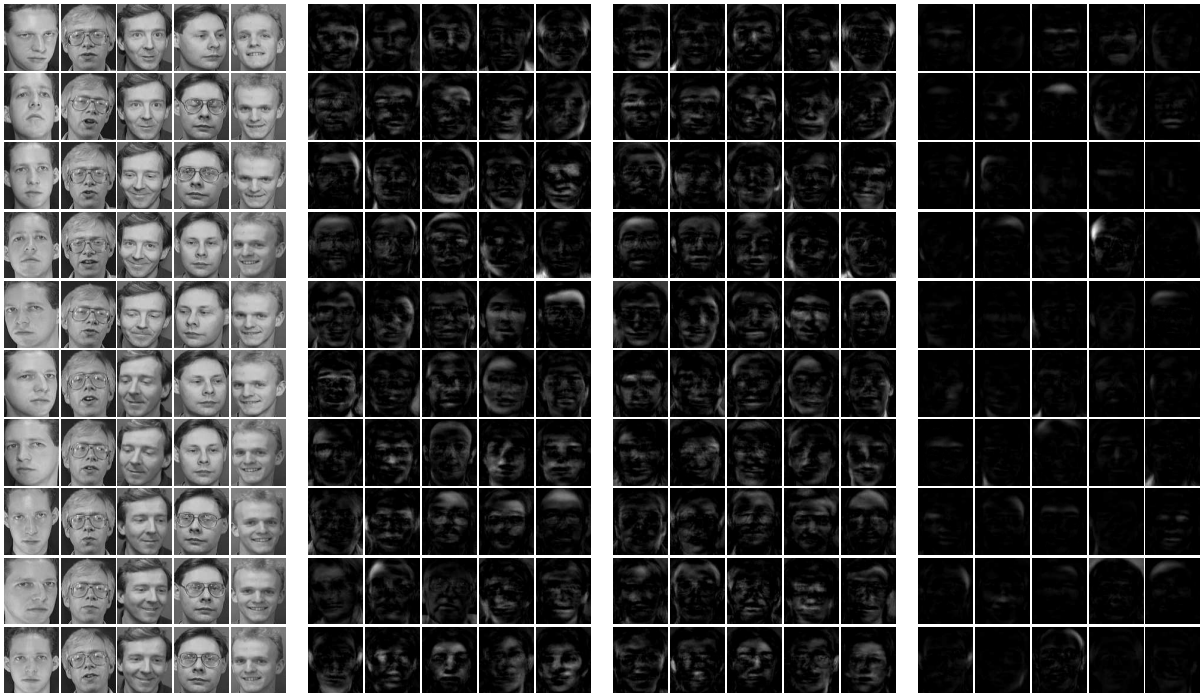
TABLE 3.3
Comparison results on all images from ORL face database with 10304×400 *tested matrix and dimension* r *varying among* $\{30, 60, 90\}$

	AAPG		ADM		blockpivot	
r	Relerr	Time	Relerr	Time	Relerr	Time
30	1.67e-1	15.8	1.71e-1	46.5	1.66e-1	74.3
60	1.41e-1	42.7	1.45e-1	88.0	1.40e-1	178
90	1.26e-1	76.4	1.30e-1	127	1.25e-1	253

ORL database has 400 images which are divided into 40 groups. Each group has 10 images of one face photoed from 10 different directions and with different expressions. We use all these images for test. Each image has 112×92 pixels and we form each one as a long column vector by stacking all of its columns. Thus a marix M of size 10304×400 is obtained. Figure 3.2 (a) depicts 50 photos corresponding to the first 50 columns of M. Again, we choose r from $\{30, 60, 90\}$. Average results of 10 independent trials are listed in Table 3.3. It demonstrates again that AAPG is better than ADM in both speed and solution quality, and AAPG is about 3 times faster than blockpivot at the expense of less than 1% accuracy. Figure 3.2 (b)-(d) depict 50 base photos for each algorithm at $r = 90$ corresponding to the first 50 columns of basis matrix X. This time, all the three algorithms get the frames of face images instead of local features. None of them get a sparse basis matrix X. The sparsest one obtained by blockpivot has 60.6% non-zeros after setting all elements below 10^{-6} to zero.

3.4. Hyperspectral data. It is shown in [18] that NMF (1.1) can be employed to spectral data analysis. A regularized NMF model is also considered in that paper with penalty terms $\alpha\|X\|_F^2$ and $\beta\|F\|_F^2$

FIG. 3.2. *ORL database and comparison base images: (a) 50 images selected from 400 tested images; (b)-(d) 50 base images corresponding to the first 50 columns of basis matrix X obtained from algorithm AAPG, ADM and blockpivot at $r = 90$.*



(a) ORL data (b) AAPG (c) ADM (d) blockpivot

FIG. 3.3. *Hyperspectral data: four selected slices.*

added in the objective function. The parameters α and β can be tuned for specific purpose in practice. However, we only focus on the original NMF model (1.1) in this section and our purpose is to show the effectiveness of our algorithm. It is worth mentioning that our method can easily be extended to such a regularized NMF model. As shown in the last section, our method works quite well for sparse-NMF, with sparse regularized terms $\alpha\|X\|_1$ and $\beta\|Y\|_1$ added in the objective function. In this subsection, we will use a $150 \times 150 \times 163$ hyperspectral cubic data to test the compared algorithms for solving (1.1). Each slice is reshaped as a long column vector by stacking its columns and the cube is scaled such that the maximum element equals to *one*. In this way, a 22500×163 matrix M is obtained. Four selected slices are shown in Figure 3.3 corresponding to the 1st, 50th, 100th and 150th columns of M. The dimension r is chosen from $\{20, 30, 40, 50\}$ for test. Average results of 10 independent trials are shown in Table 3.4. We can see from the table that AAPG is superior to ADM and blockpivot in both time and solution quality.

4. Conclusion and discussions. This paper proposed an alternating proximal gradient method for solving nonnegative matrix factorization problem. Under some mild assumptions, it is proved that any

TABLE 3.4
Comparison results on hyperspectral cube of size $150 \times 150 \times 163$ *with dimension* r *varying among* $\{20, 30, 40, 50\}$

	AAPG		ADM		blockpivot	
r	Relerr	Time	Relerr	Time	Relerr	Time
20	1.18e-2	34.2	2.34e-2	87.5	1.38e-2	62.5
30	9.07e-3	63.2	2.02e-2	116	1.10e-2	143
40	7.56e-3	86.2	1.78e-2	140	9.59e-3	194
50	6.45e-3	120	1.58e-2	182	8.e-3	277

limit point of the iterates generated by this method satisfies first-order optimality conditions. However, this method converges quite slow as the iterate approaches to the solution. Inspired by the fact that a Nesterov-type acceleration technique can significantly speed up the proximal gradient method for convex program, we applied this extrapolation technique to the non-convex problem. It turns out that the extrapolation helps a lot to accelerate the proposed method. Extensive numerical experiments illustrate the effectiveness of the accelerated alternating proximal gradient method (AAPG). The test on synthetic data shows that AAPG is comparable with other state-of-the-art algorithms in both speed and solution quality. All tests on real data demonstrate that AAPG has a significant advantage in speed with comparable accuracy as other methods. Especially for the hyperspectral data test, AAPG shows its superiority in both speed and solution quality among the compared algorithms.

Before we finish this paper, we would like to show that the accelerated alternating proximal gradient method can be easily applied to the following sparse-NMF model

$$\min_{X,Y\geq 0} \frac{1}{2}\|XY - M\|_F^2 + \alpha\|X\|_1 + \beta\|Y\|_1, \tag{4.1}$$

where $\|X\|_1 = \sum_{ij} |X_{ij}|$. Due to nonnegativity, $\|X\|_1 = \sum_{ij} X_{ij}$. Just as how we derived APG in Section 2, we can alternatively approximate the objective function of (4.1) by a quadratic function and then minimize the quadratic approximation function with respect to X and Y at each iteration. Similar to (2.8), all the updates can be written in a closed form as follows.

$$t_k = (1 + \sqrt{1 + 4t_{k-1}^2})/2 \tag{4.2a}$$
$$\omega_k = (t_{k-1} - 1)/t_k \tag{4.2b}$$
$$Z_X^k = X^k + \omega_k(X^k - X^{k-1}) \tag{4.2c}$$
$$Z_Y^k = Y^k + \omega_k(Y^k - Y^{k-1}) \tag{4.2d}$$
$$X^{k+1} = \max(0, Z_X^k - (\nabla_X F(Z^k, Y^k) + \alpha)/L_{X^k}), \tag{4.2e}$$
$$Y^{k+1} = \max(0, Z_Y^k - (\nabla_Y F(X^{k+1}, Z_Y^k) + \beta)/L_{Y^k}), \tag{4.2f}$$

where F and L_{X^k}, L_{Y^k} have the same meanings as in Section 2. The method (4.2) is named as algorithm SAAPG. We compared algorithm AAPG, SAAPG, ADM and blockpivot on 500×500 random matrices with sparse basis X. Each matrix can be written as $M = LR$, where $L \in \mathbb{R}_+^{m\times r}$ is generated by calling Matlab command `sprandn` with density argument $p = 0.1, 0.15, 0.20$ and $R \in \mathbb{R}_+^{r\times n}$ is generated by calling `randn`. Parameters α and β are set to $0.025p$ and 0 for SAAPG, respectively. All other settings are the same as in Section 3. Table 4.1 lists the comparison results, all of which are average of 10 independent trials. Relerr and time have the same meaning as in Section 3. Density denotes the percentage of non-zeros in the basis matrix X after setting elements below 10^{-6} to zero. Though the density parameter is set to p, the actual percentage of non-zeros of matrix L is usually less than p. That is why SAAPG and blockpivot can get sparser basis matrix X than the specified density level p. We can see from the table that all algorithms get high-quality solutions, except SAAPG and blockpivot failed once when $r = 30, p = 0.10$. With the sparse regularized term, SAAPG indeed obtains sparser solution than what AAPG gets without the penalty term and SAAPG gets sparsest basis in average for all cases.

REFERENCES

TABLE 4.1
Comparison results on 500×500 *Gaussian random matrices with sparse basis*

	AAPG			SAAPG			ADM			blockpivot		
(r, p)	Relerr	Time	Density	Relerr	Time	Density	Relerr	Time	Density	Relerr	Time	Density
(10, 0.10)	6.06e-5	0.16	0.1515	5.72e-5	0.17	0.0702	3.39e-5	0.15	0.2645	1.52e-5	0.45	0.0951
(10, 0.15)	5.60e-5	0.17	0.1879	5.87e-5	0.18	0.0924	3.88e-5	0.12	0.4351	1.36e-5	0.41	0.1342
(10, 0.20)	5.77e-5	0.16	0.2691	5.71e-5	0.17	0.1126	2.74e-5	0.12	0.5053	1.72e-5	0.46	0.1732
(20, 0.10)	7.07e-5	0.40	0.1525	7.25e-5	0.40	0.0694	8.39e-5	0.26	0.5496	3.45e-5	1.03	0.1071
(20, 0.15)	6.79e-5	0.38	0.2151	6.94e-5	0.40	0.0981	9.34e-5	0.25	0.6846	3.51e-5	1.12	0.1428
(20, 0.20)	7.02e-5	0.42	0.2647	6.91e-5	0.41	0.1195	9.93e-5	0.25	0.8223	3.53e-5	1.29	0.1610
(30, 0.10)	8.07e-5	0.84	0.1342	4.93e-3	0.77	0.0809	2.28e-4	0.51	0.7098	5.06e-3	1.65	0.1097
(30, 0.15)	8.04e-5	0.77	0.1913	7.77e-5	0.83	0.0987	9.38e-5	0.46	0.7720	4.41e-5	1.99	0.1306
(30, 0.20)	7.76e-5	0.79	0.2640	6.98e-5	0.81	0.1262	7.37e-5	0.44	0.9043	5.90e-5	2.09	0.1522

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[3] Y. Chen, M. Rege, M. Dong, and J. Hua. Non-negative matrix factorization for semi-supervised data clustering. *Knowledge and Information Systems*, 17(3):355–379, 2008.

[4] A. Cichocki, M. Morup, P. Smaragdis, W. Wang, and R. Zdunek. Advances in nonnegative matrix and tensor factorization. *Computational intelligence and neuroscience*, 2008.

[5] A. Cichocki, R. Zdunek, and A.H. Phan. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley, 2009.

[6] C. Ding, T. Li, and M.I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 45–55, 2008.

[7] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[8] H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. *SIAM J. Matrix Anal. Appl*, 30(2):713–730, 2008.

[9] J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 353–362. IEEE, 2008.

[10] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[11] D.D. Lee and H.S. Seung. Algorithms for Non-Negative Matrix Factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[12] S.Z. Li, X.W. Hou, H.J. Zhang, and Q.S. Cheng. Learning spatially localized, parts-based representation. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–207. IEEE, 2001.

[13] C.J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[14] Q. Ling, Y. Xu, W. Yin, and Z. Wen. Decentralized low-rank matrix completion. *Rice Technical Report*, 2011.

[15] Y. Nesterov. A method of solving a convex programming problem with convergence ratio$o(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[16] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[17] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[18] V.P. Pauca, J. Piper, and R.J. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and its Applications*, 416(1):29–47, 2006.

[19] V.P. Pauca, F. Shahnaz, M.W. Berry, and R.J. Plemmons. Text mining using nonnegative matrix factorizations. In *Proc. SIAM Inter. Conf. on Data Mining, Orlando, FL*, 2004.

[20] C. Thurau, K. Kersting, and C. Bauckhage. Convex non-negative matrix factorization in the wild. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 523–532. IEEE, 2009.

[21] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Rice Technical Report*, 2010.

[22] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

[23] Y. Zhang. An alternating direction algorithm for nonnegative matrix factorization. *Rice Technical Report*, 2010.